CS800, Practice Final

NOTE: This file contains more than five problems – it is a collection of several final-exam-like problems.

Name: _________________________

- This exam is worth 40 points. It consists of five problems, each worth 10 points. The sum of the four highest scored problems defines the final grade.

- If you want to ask a question, write it on the provided sheet of paper and raise your hand. I will collect it, write down the answer and return it back to you.

- If you need more scrap paper, raise your hand. Use only the scrap paper provided.

- Please turn off your cell-phones and other electronic devices.

- When time is called, stop writing immediately and remain seated until all exams have been collected.

- Hand in also your scrap paper and cheat sheet. Stop by my office next quarter to get your cheat sheet back.
Problem 1

Given is a tree $T = (V, E)$. Give a linear-time algorithm which determines whether $T$ has a perfect matching.
Recall: A matching is a collection of edges that do not share vertices. A matching is perfect if every vertex is covered by an edge from the matching.

a) Give the pseudocode of your algorithm. You may use any algorithms from class as subroutines.

b) Briefly explain your algorithm.

c) Argue your algorithm’s running time.
Problem 2

Given is a set of $n$ four-letter words $w_1, w_2, \ldots, w_n$ (all spelled in lower-case letters). Given $i, j$, we need to change $w_i$ to $w_j$ by changing one letter at a time or permuting the letters in the word and along the way we have to use only words from \{ $w_1, w_2, \ldots, w_n$ \}.

For example, for words “ahoy”, “algo”, “foal”, “goal”, “good”, “leaf”, “leap”, “loaf”, “luck”, “pale”, “plea”, we can get from “algo” to “leaf” as follows: “algo”, “goal”, “foal”, “loaf”, “leaf”. We cannot get from “good” to “luck”.

Give an $O(n)$ algorithm that outputs POSSIBLE if it is possible to get from $w_i$ to $w_j$, otherwise it outputs impossible.

a) Give the pseudocode of your algorithm. You may use any algorithms from class as subroutines.

b) Briefly explain your algorithm.

c) Argue your algorithm’s running time.
Problem 3

a) Given is the following weighted graph $G$. Trace the Dijkstra’s algorithm to compute the shortest distances from vertex $s$ to every other vertex. For every vertex mark the shortest path leading to it.

![Graph](image)

b) Find the minimum spanning tree of $G$ (mark it on the graph below).

![Graph](image)

c) Given is the following simple algorithm:

1. Let $T$ be the minimum spanning tree of $G$.
2. For every vertex $v \in V$ do
3. Compute the length of the path from $s$ to $v$ along the edges of $T$.
4. Output the computed lengths as the shortest distances from $s$ to all other vertices.

Does the algorithm work? Argue your answer.
Problem 4

Given is a computer network with \(n\) computers. Given are also \(m\) direct connections specifying pairs of connected computers and for each connection we know the bandwidth (i.e. how many gigabytes per second can be sent through the connection). Two of the computers, \(s\) and \(t\), are very important and a lot of data is being sent from \(s\) to \(t\). We want to invest and increase the bandwidth on one of the connections so that we can send more gigabytes per second from \(s\) to \(t\).

Design an \(O(nm(n + m))\) algorithm that finds one connection such that increasing the bandwidth on the connection would increase the number of gigabytes per second that can be sent from \(s\) to \(t\).

a) Give the pseudocode of your algorithm. You may use any algorithms from class as subroutines.

b) Briefly explain your algorithm.

c) Argue your algorithm’s running time.
Problem 5

Recall that a clique of size $\ell$ in an undirected graph is a set of $\ell$ vertices such that every pair of these vertices is connected by an edge.

Also recall the CLIQUE problem where, given an undirected graph $G$ and a number $k$, the answer is YES if and only if $G$ contains a clique of size $k$. This problem is known to be NP-complete.

Consider the following problems:

A. Given a graph $G$ and a number $k$, does there exist a clique of size $k$?

B. Given a graph $G$ and a number $k$, does there exist a clique of size smaller than $k$?

C. Given a graph $G$ and a number $k$, does there exist a clique of size larger than $k$?

D. Given a graph $G$, does there exist a clique of size 5?

E. Given a graph $G$, does there exist a clique of size $n - 5$ (where $n$ is the number of vertices)?

F. Given a graph $G$, does there exist a clique of size $n/2$ (where $n$ is the number of vertices)?

G. Given an acyclic graph $G$ and a number $k$, does there exist a clique of size $k$?

H. Given a graph $G$, is the largest clique of size $k$?

I. Given a graph $G$, is the smallest clique of size $k$?

List all the problems that are in P, all the problems that are in NP, and all the problems that are NP-complete. You do not have to give any reasoning.
Problem 6

The BINPACKING problem is defined as follows. Given are positive integers \( n, k, B \), and a sequence of positive integers \( w_1, w_2, \ldots, w_n \). The number \( n \) represents \( n \) items numbered 1, 2, \ldots, \( n \), where the \( i \)-th object weights \( w_i \) pounds, for every \( i = 1, \ldots, n \). The number \( k \) represents \( k \) identical boxes, each capable of holding \( B \) pounds. The BINPACKING problem decides if it is possible to put every item in a box so that no box holds more than \( B \) pounds. This problem is known to be NP-complete.

(a) Your friend decided to try the following algorithm:

1. Sort the items by decreasing weight.
2. For \( b = 1 \) to \( k \) let \( \text{RemainingCapacity}[b] = B \).
3. For \( i = 1 \) to \( n \) do
4. Find bin \( b \) with the smallest \( \text{RemainingCapacity} \) that is at least \( w_i \).
5. If there exists such \( b \) then
6. Place the \( i \)-th item in the bin \( b \).
7. Set \( \text{RemainingCapacity}[b] = \text{RemainingCapacity}[b] - w_i \).
8. Else Return IMPOSSIBLE.
9. Return POSSIBLE.

Is the algorithm likely to work? Reason your answer.

(b) Reason that BINPACKING is in NP.
Problem 7

Given is a sequence of numbers $a_1, a_2, \ldots, a_n$. Cut the sequence into the minimum possible number of contiguous subsequences such that each subsequence starts and ends with the same number.

Example: Sequence $8, 2, 6, 7, 1, 2, 3, 4, 5, 4, 5, 3, 1$ can be cut into four contiguous subsequences that start and end with the same number as follows: $8 | 2, 6, 7, 1, 2 | 3, 4, 5, 4, 5, 3 | 1$. It is not possible to cut the sequence into fewer such subsequences.

Give an $O(n^2)$ algorithm that computes the minimum number of such subsequences. Use dynamic programming with the following first part of the heart:

\[
S[j] = \text{the minimum number of contiguous subsequences that start and end with the same number that the sequence } a_1, a_2, \ldots, a_j \text{ can be cut into}
\]

Give the rest of the heart:

- The mathematical expression of $S[j]$:

- The return value of the algorithm:

Pseudocode of your algorithm: